FIG. 1

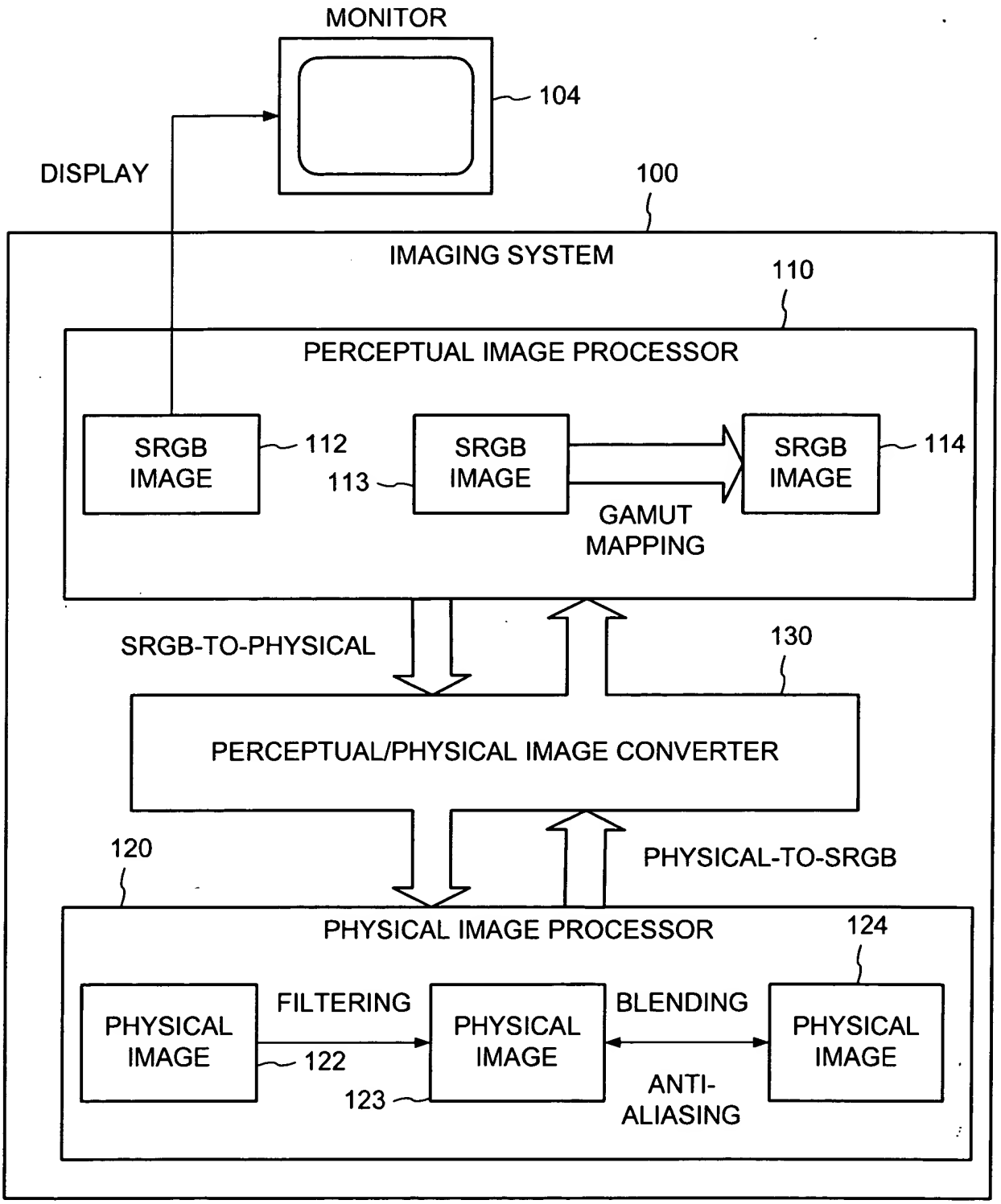


FIG. 2

200

```
extern "C" void
sRGBColor( Color *pOut, Color *pIn )
{
   for ( int i = 0; i < 4; i++)
   {
      float x = (*pIn)[i];
      if ( x < 0.03928f )
         x = x/12.92f;
      else
         x = powf(((x + 0.055f)/1.055f), 2.4f);
      (*pOut)[i] = x;
   }
}
```

FIG. 3

300

```
extern "C" void
LinearTosRGBColor( Color *pOut, Color *pIn )
{
   // coarse approximation: weighted arithmetic mean between
   // x^0.5 and x^0.375 approximates x^(1/2.4)
   for ( int i = 0; i < 4; i++)
   {
      float x = (*pIn)[i];
      float sqrtx = sqrtf(x);
      float sqrt3x = sqrtf(sqrtf(sqrtx));
      float pow124 = 0.38f*sqrtx+0.62f*sqrtx/sqrt3x;
      if ( x < 0.00304f )
         x = 12.92f * x;
        else
           x = 1.055f*pow124-0.055f;
        (*pOut)[i] = x;
   }
}
```

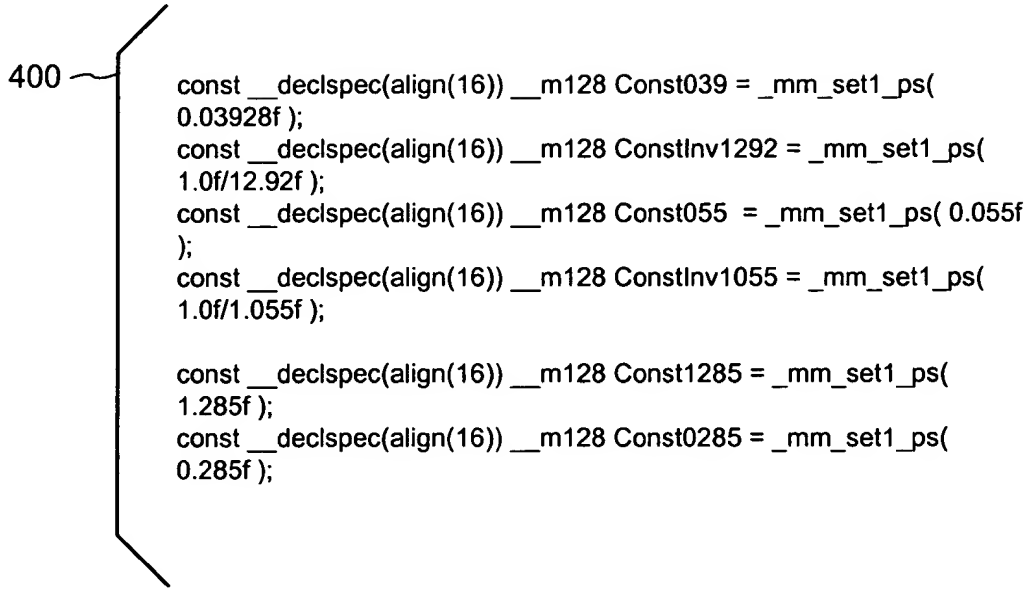
FIG. 4

310

```
extern "C" void
LinearTosRGBColor( Color *pOut, Color *pIn )
{
   // finer approximation that avoids taking 3 successive
   // square roots: apply one round of N-R to guess cube
   // root of x*sqrt(sqrt(x))
   for ( int i = 0; i < 4; i++)
   {
      float x = (*pIn)[i];

      float sqrtx = sqrtf(x);
      float sqrt2x = sqrtf(sqrtx);
      float appx = 0.78f*sqrtx+0.22f*sqrt2x;
      float num = x*sqrt2x;
      float cuberoot = (2*appx+(x*sqrt2x)/(appx*appx))/3.0f - 0.00025f;
      if ( x < 0.00304f )
         x = 12.92f * x;
        else
           x = 1.055f*cuberoot-0.055f;
        (*pOut)[i] = x;
   }
}
```

FIG. 5



401

FIG. 6

```
extern "C" void
sRGBColor( Color *pOut, Color *pIn )
{
   // SIMD: compute BOTH answers and compose output using mask
   __m128 ansBelowDelta = _mm_mul_ps( *(__m128 *) pIn, ConstInv1292 );
   __m128 x = _mm_mul_ps( ConstInv1055, _mm_add_ps( *(__m128 *) pIn,
Const055 ) );
   __m128 sqrx = _mm_mul_ps( x, x );
   __m128 invsqrx = _mm_rcp_ps( sqrx );
   __m128 invsqrtx = _mm_rsqrt_ps( x );
   __m128 ansAboveDelta = _mm_div_ps( Const1285,
                  _mm_mul_ps( invsqrx, _mm_add_ps( Const0285, invsqrtx ) ) );
   __m128 TrueIfLTDelta = _mm_cmplt_ps( *(__m128 *) pIn, Const039 );
   *(__m128 *) pOut = _mm_or_ps( _mm_and_ps( TrueIfLTDelta, ansBelowDelta
),
                        _mm_andnot_ps( TrueIfLTDelta, ansAboveDelta ) );
}
```

500

FIG. 7

```
const __declspec(align(16)) __m128 CONST00304 = _mm_set1_ps( 0.00304f );
const __declspec(align(16)) __m128 CONST1292 = _mm_set1_ps( 12.92f );
const __declspec(align(16)) __m128 CONST055  = _mm_set1_ps( 0.055f );
const __declspec(align(16)) __m128 CONST1055 = _mm_set1_ps( 1.055f );

const __declspec(align(16)) __m128 CONST078 = _mm_set1_ps( 0.78f );
const __declspec(align(16)) __m128 CONST1m078 = _mm_set1_ps( 1.0f-0.78f );

const __declspec(align(16)) __m128 CONST38 = _mm_set1_ps( 0.38f );
const __declspec(align(16)) __m128 CONST1m38 = _mm_set1_ps( 1.0f-0.38f );

extern "C" void
LinearTosRGBColor( Color *pOut, Color *pIn )
{
   __m128 ansBelowDelta = _mm_mul_ps( *(__m128 *) pIn, CONST1292 );
   __m128 sqrtx = _mm_sqrt_ps( *(__m128 *) pIn );
   __m128 sqrt3x = _mm_sqrt_ps( _mm_sqrt_ps( sqrtx ) );
   __m128 pow124 = _mm_add_ps( _mm_mul_ps( CONST38, sqrtx ),
                       _mm_div_ps( _mm_mul_ps( CONST1m38, sqrtx ), sqrt3x ) );
   __m128 ansAboveDelta = _mm_sub_ps( _mm_mul_ps( CONST1055, pow124 ), CONST055 );
   __m128 TrueIfLTDelta = _mm_cmplt_ps( *(__m128 *) pIn, CONST00304 );
   *(__m128 *) pOut = _mm_or_ps( _mm_and_ps( TrueIfLTDelta, ansBelowDelta ),
                       _mm_andnot_ps( TrueIfLTDelta, ansAboveDelta ) );
}
```

500

FIG. 8

```
const __declspec(align(16)) __m128 Magic00304 = _mm_set1_ps( 0.00304f );
const __declspec(align(16)) __m128 Magic1292 = _mm_set1_ps( 12.92f );
const __declspec(align(16)) __m128 Magic055  = _mm_set1_ps( 0.055f );
const __declspec(align(16)) __m128 Magic1055 = _mm_set1_ps( 1.055f );
const __declspec(align(16)) __m128 MagicInv3 = _mm_set1_ps( 1.0f/3.0f );
const __declspec(align(16)) __m128 MagicFudge = _mm_set1_ps( 0.00025f );

const __declspec(align(16)) __m128 Magic078 = _mm_set1_ps( 0.78f );
const __declspec(align(16)) __m128 Magic1m078 = _mm_set1_ps( 1.0f-0.78f );

const __declspec(align(16)) __m128 Magic38 = _mm_set1_ps( 0.38f );
const __declspec(align(16)) __m128 Magic1m38 = _mm_set1_ps( 1.0f-0.38f );

extern "C" void
LinearTosRGBColor( Color *pOut, Color *pIn )
{
   __m128 ansBelowDelta = _mm_mul_ps( *(__m128 *) pIn, Magic1292 );
   __m128 sqrtx = _mm_sqrt_ps( *(__m128 *) pIn );
   __m128 sqrt2x = _mm_sqrt_ps( sqrtx );
   __m128 appx = _mm_add_ps( _mm_mul_ps( Magic078, sqrtx ),
                     _mm_mul_ps( Magic1m078, sqrt2x ) );
   __m128 cuberoot = _mm_sub_ps(
                  _mm_mul_ps( MagicInv3,
                     _mm_add_ps( _mm_add_ps( appx, appx ),
                        _mm_div_ps( _mm_mul_ps( *(__m128 *) pIn, sqrt2x ),
                           _mm_mul_ps( appx, appx )
                        )
                     )
                  ), MagicFudge
               );
   __m128 ansAboveDelta = _mm_sub_ps( _mm_mul_ps( Magic1055, cuberoot ), Magic055 );
   __m128 TrueIfLTDelta = _mm_cmplt_ps( *(__m128 *) pIn, Magic00304 );
   *(__m128 *) pOut = _mm_or_ps( _mm_and_ps( TrueIfLTDelta, ansBelowDelta ),
                        _mm_andnot_ps( TrueIfLTDelta, ansAboveDelta ) );
}
```

FIG. 9